Operating Systems (2INC0)

Scheduling

Dr. Geoffrey Nelissen

Courtesy of Prof. Dr. Johan Lukkien and Dr. Tanir Ozcelebi

IRiS
Interconnected
Resource-aware
Intelligent Systems

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

A single process (thread) may not use the system resources efficiently

Only **one task** can use a resource (e.g. processor, memory page, device) **at any given time instant**.

A task **does not use a resource constantly** during its execution

→ We **must schedule** the access to resources

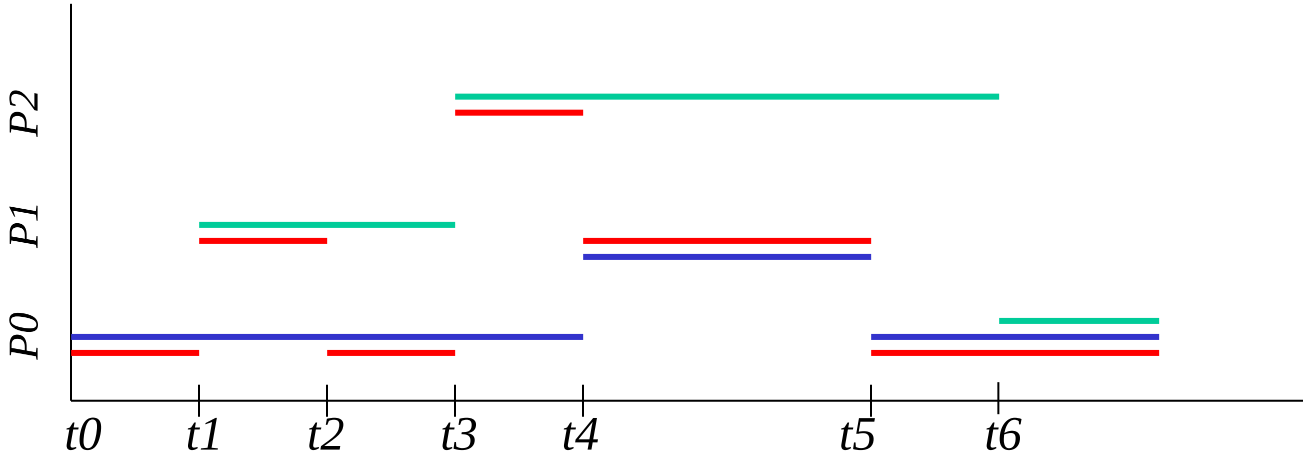
Agenda

- Resource scheduling
- CPU scheduling
- Common scheduling algorithms
- Priority inversion

Resource scheduling (allocation)

- Assignment of resources to tasks
  - **Schedule *S*** is a **function that maps a time and a resource to a *task*:** $S(t,r) = P$ means that task P *is* allocated resource r at time t
  - Definition of a ***task*** is context dependent
    - e.g. a **process**, a **thread** of execution, **I/O operation** (e.g., the reading of a disk block), servicing an **interrupt**, etc.
  - To define a schedule we must decide
    - ***When:*** to change the allocation of a resource
      - Usually when there is a **change in the system state**
        - processor: e.g. process added into ready queue; end of time slice; process yielding (e.g. blocked, waiting for I/O or for child process to terminate)
        - memory: e.g. memory management call; replacement policy by memory subsystem; process termination
    - ***How:*** what decision *procedures* are used to allocate the resource when in decision mode

Example schedule



- *P0, P1, P2*: tasks (processes, jobs)
- *t0, t1*,....: scheduling points
  - system is in decision mode
  - scheduling decisions are taken
  - scheduling points may differ per resource (not shown here)

- processor resource: *PROC*
  - *S([t0..t1), PROC) = P0,*
    S([t1..t2), PROC) = P1, etc.
- a memory page frame: *m23*
  - *S([t0..t4), m23) = P0*, etc.
- another memory page frame: *m56*
  - *S([t3..t6), m56) = P2, etc.*

Scheduling policies and mechanisms

- ***Scheduling policy*** represents the strategy for allocating a resource to a task while in *decision mode.*
  - policy: an algorithm that decides based on *scheduling criteria....*
    - task attributes (deadline, response time, ...)
    - current state (the set of ready processes, available and required resources, …)
  - ....or based on a pre-computed lookup table

Agenda

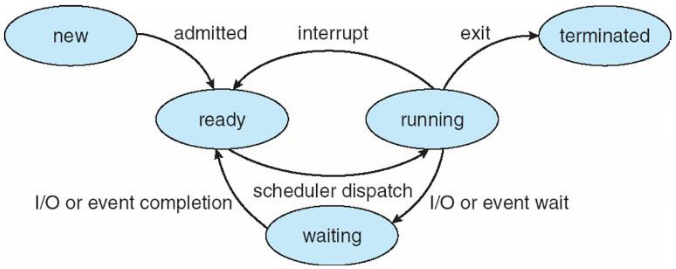
- Resource scheduling
- CPU scheduling
  - metrics, task attributes
  - a framework for scheduling
- Common scheduling algorithms
- Priority inversion

Metrics for the quality of CPU scheduling

Scheduling Criteria

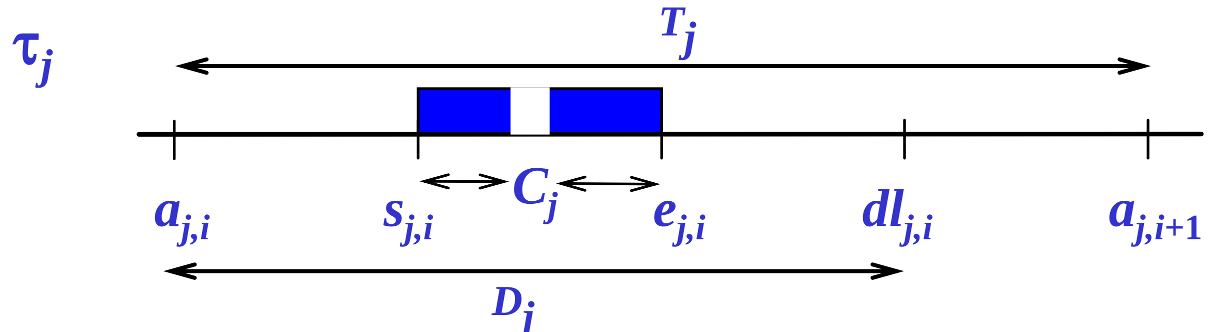
Metric: *observed* property, result of applying scheduling policy

- CPU utilization
  - keep the CPU as busy as possible
- Throughput
  - # of processes that complete their execution per time u
- number of deadline misses (real-time scheduling)
- turnaround time
  - time to execute a particular process
- waiting time
  - time a process has been waiting in the ready queue
- response time
  - time it takes from when a request was submitted until the first response is produced → not the entire output (suitable for time-sharing interactive environment)
    - most texts: *response time* is defined as the time elapsing from arrival to completion (in our textbook this is called: *turnaround time*)



Basis for our comparisons of different scheduling schemes in this slide set

Time attributes of a task



- A task has fixed time attributes and …
  - a name (the j^{th} task) — τ_j
  - a period, sometimes — T_j
  - a (worst case) execution time — C_j
  - a relative deadline, sometimes — D_j

- … dynamic time attributes (i^{th} instance or occurrence)
  - an **arrival** time — $a_{j,i}$
  - an **absolute deadline**, sometimes (add D_j to arrival time) — $dl_{j,i}$
  - a **start** time (or beginning time) of execution — $s_{j,i}$
  - a departure time, also called **end**, finish or completion time — $e_{j,i}$

- (book) Response time: $s_{j,i} - a_{j,i}$
- (book) Turnaround time: $e_{j,i} - a_{j,i}$

CPU scheduling

- Scheduling framework defines 3 things
  - **When** to schedule?
    = ***decision mode***
  - **Priority function** (priority scheduling)
    - What task to schedule based on priority?
  - **Arbitration rule**
    - who to schedule **in case of equal priority**?

- Managed by 2 different OS modules (jointly called *the scheduler*)
  - Process scheduler
    - policies to determine **which task to execute next**
  - Process dispatcher
    - actual **binding of selected task to a processor** (switching context, switching to user mode, jumping to the proper location in the program)

Decision mode: Activating process scheduler

- Decision mode defines the conditions for activating the process scheduler.
  - Upon activation
    - Task scheduler selects one ready task for execution

- Scheduler can be invoked (decision mode), e.g.:
  1. **Periodically**
  2. when a **process** switches **from running to waiting state** (e.g. wait for child to terminate)
  3. when a **process** switches **from running to ready state** (e.g. through an interrupt)
  4. when a **process** switches **from waiting to ready** (e.g. i/o completion)
  5. when a **process terminates**

- If scheduling takes place ONLY under 2 & 5 (active process voluntarily yields) → non-preemptive decision mode

  otherwise → preemptive decision mode